Published for SISSA by Springer

Received: March 3, 2020
Revised: April 15, 2020
Accepted: May 26, 2020
Published: June 17, 2020

Using neural networks for efficient evaluation of high multiplicity scattering amplitudes

Simon Badger[a] **and Joseph Bullock**[a,b]

[a] *Institute for Particle Physics Phenomenology, Department of Physics, Durham University, Durham DH1 3LE, U.K.*
[b] *Institute for Data Science, Durham University, Durham DH1 3LE, U.K.*

E-mail: simon.d.badger@durham.ac.uk, j.p.bullock@durham.ac.uk

Abstract: Precision theoretical predictions for high multiplicity scattering rely on the evaluation of increasingly complicated scattering amplitudes which come with an extremely high CPU cost. For state-of-the-art processes this can cause technical bottlenecks in the production of fully differential distributions. In this article we explore the possibility of using neural networks to approximate multi-variable scattering amplitudes and provide efficient inputs for Monte Carlo integration. We focus on QCD corrections to $e^+e^- \to$ jets up to one-loop and up to five jets. We demonstrate reliable interpolation when a series of networks are trained to amplitudes that have been divided into sectors defined by their infrared singularity structure. Complete simulations for one-loop distributions show speed improvements of at least an order of magnitude over a standard approach.

Keywords: NLO Computations, QCD Phenomenology

ArXiv ePrint: 2002.07516

https://doi.org/10.1007/JHEP06(2020)114

JHEP06(2020)114

Contents

1 Introduction

Improvements in the precision high energy collider experiments are putting increasing pressure on theoretical predictions. The latest technology for the evaluation of scattering amplitudes, the handling of infrared singularities and Monte Carlo event generation has been able to achieve an impressive range of predictions for differential observables at NLO and NNLO in both QCD and EW coupling expansions. Despite the successes, simulations at the cutting edge of the precision frontier are often extremely computationally expensive.

In this article we explore one way in which popular computer science technology can be used to decrease the computational cost of precision simulations. In particular, we consider high multiplicity scattering processes, with extremely high mathematical complexity, where it is less clear how to make use of conventional interpolation methods such as polynomial fits and interpolation grids [1–5]. Neural networks, however, are by now a standard tool within the data analysis, data science and machine learning communities and offer a general, non-linear parametrisation which have the potential to approximate any continuous function [6], and therefore could be useful in the context of high multiplicity scattering.

The basic principle is not new of course. Neural networks have the potential to provide extremely fast and lightweight approximations of complicated amplitudes. In figure 1 we demonstrate this for the particular test cases which are the subject of this article, the tree-level and one-loop amplitudes inside the NJET amplitude generator [7] for $e^+e^- \to \leq 5$

JHEP06(2020)114

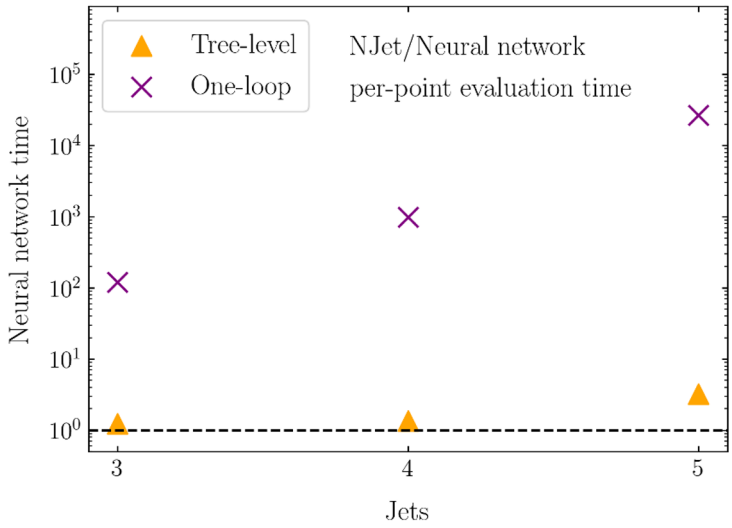

Figure 1. A comparison of the CPU cost of tree-level and one-loop amplitudes to a generic neural network (Keras/TensorFlow) as a function of the number of legs (equivalently number of variables). This demonstrates the very obvious fact that the neural network is fast to call and has a very mild dependence on the number of variables. The challenge is to train the network well enough that it can be interpolated and extrapolated reliably over a complete range of differential observables.

jets. While the potential speed up in the function call is quite striking, the real challenge is not clear from this analysis. The actual improvement in CPU cost must take into account the time taken to train the network to a level that it can be interpolated and extrapolated accurately and reliably.

Previous attempts have been made to use machine learning tools such as Boosted Decision Trees (BDTs) and neural networks for efficient phase-space sampling and Monte Carlo integration [8, 9] with recent work [10, 11] focusing on the use of coupling layers [12]. Similarly, work such as that of Otten et al. [13] makes use of neural networks for explicit cross-section prediction. Here, the authors focus on $pp \to 2$ jet processes, and implement an Artificial Neural Network point selection (NNPS) scheme for selecting training data based on the points the network struggles to learn the most. In addition, there has been much work on the use of Generative Adversarial Networks (GANs) [14], and other generative models, for event generation [15–21], while there has been little work addressing the issue of explicit matrix element approximation [22].

In this paper we design a deep learning pipeline to approximate $e^+e^- \to \leq 5$ jet matrix elements at both LO and NLO, thus exploring processes with significantly higher multiplicity than those previously considered. While [13] uses a more automated approach for phase-space sampling to aid in training a neural network, we employ physics-based knowledge of the processes in designing our pipeline and analyse the effectiveness of this approach and what this might tell us about the phenomenological set up. We pay careful attention to the errors and uncertainties in our neural network approximation, and offer a comprehensive implementation of neural network regression analysis.

For usability, we supply code to accompany our methodology and results [23].

JHEP06(2020)114

2 Computational setup

We use the on-shell based C++ code NJET [7], to evaluate colour and helicity summed Born and virtual matrix elements for $e^+e^- \to \leq 5$ jets, denoted $\mathcal{M}^{(n,0)}$ and $\mathcal{M}^{(n,1)}$ respectively. NJET uses integrand level reduction [24] and generalised unitarity [25–31] to construct loop amplitudes from tree-level input computed efficiently with Berends-Giele recursion [32]. For a given phase-space point, NJET calculates the virtual and Born matrix elements, along with the $1/\epsilon$ and $1/\epsilon^2$ correction coefficients, from which we can calculate the k-factors:

$$\text{k-factor} = \frac{\mathcal{M}^{(n,1)}}{\mathcal{M}^{(n,0)}}. \tag{2.1}$$

For ease of use, NJET is interfaced with via the Binoth Les Houches Accord (BLHA) [33], which is designed to provide a standardised interface between Monte Carlo tools and matrix element programs.

We explore the performance of various neural network parameterisations of the amplitude for total and differential cross-section computations at LO, as well as their corresponding k-factor equivalents at NLO. We find that as the multiplicity increases, infrared singularities on the edge of the phase-space increasingly cause problems for a single neural network, which struggles to find a good fit across the whole phase-space. To improve the approximation, we divide up the phase-space into sectors according to the subtraction method developed by Frixione, Kunszt and Signer (FKS) [34, 35]. Although we do not actually perform subtraction, this phase-space decomposition isolates the infrared singularities and allows the training of networks focused on improving their performance on each partition individually.

2.1 Phase-space partitioning

We explore two pipeline configurations: i) we naively train a single network over all sampled points in phase-space; ii) we divide the phase-space into divergent and non-divergent regions in an attempt to partially isolate the infrared singularities and then further sub-divide the divergent region according to the FKS subtraction method, training one network on the non-divergent region, and a different network on each partition. For clarity we will generally refer to the naive single network, and partitioned ensemble of networks, as 'models' and the individual networks comprising these models as 'networks'.

We parameterise our phase-space according to the Lorentz invariant $y_{ij} = s_{ij}/s_{\text{com}}$, where $s_{ij} = (p_i + p_j)^2$ and s_{com} is the centre-of-mass energy of the incoming particles, and define all cuts with respect to this quantity. Let the global phase-space cuts be denoted y_{cut} and the partition dividing divergent and non-divergent regions be at y_p. Using these two scales, the divergent region, $\mathcal{R}_{\text{div}}$, and the non-divergent region, $\mathcal{R}_{\text{non-div}}$, are defined as follows:

$$\mathcal{R}_{\text{div}} = \{p \,|\, y_{\text{cut}} \leq \min(y_{ij}) \leq y_{\text{cut}} + y_p,\ p = (p_a, p_b, p_1, \ldots, p_n),\ i, j \in \{1, \ldots, n\}\}, \tag{2.2}$$

$$\mathcal{R}_{\text{non-div}} = \{p \,|\, y_{\text{cut}} + y_p \leq \min(y_{ij}),\ p = (p_a, p_b, p_1, \ldots, p_n),\ i, j \in \{1, \ldots, n\}\}, \tag{2.3}$$

JHEP06(2020)114

where p is a phase-space point consisting of the initial state 4-momenta, p_a and p_b, and the outgoing momenta, $\{p_1, p_2, \ldots, p_n\}$, where n is the number of jets.

In the FKS subtraction formalism, the phase-space is divided such that the kinematic regions resulting from each partition contain only a specific subset of singularities. In order to achieve this, a set of ordered pairs, known as FKS pairs, are introduced. In our case of $e^+e^- \to \leq 5$ jets we define these as:

$$\mathcal{P}_{\text{FKS}} = \{(i,j) \,|\, 1 \leq \ i \leq n_g + 2,\ 3 \leq \ j \leq n_g + 2, i \neq j, \\ \mathcal{M}^{(n,0)} \text{ or } \mathcal{M}^{(n,1)} \to \infty \text{ if } p_i^0 \to 0 \text{ or } p_j^0 \to 0 \text{ or } \vec{p}_i || \vec{p}_j\}, \quad (2.4)$$

where n_g is the number of gluons in the process.

We then construct a partition function similar to that of [36, 37] (for a brief introduction to different FKS pair definitions and partition choices see appendix B):

$$\mathcal{S}_{i,j} = \frac{1}{D_1 s_{ij}}, \quad D_1 = \sum_{i,j \in \mathcal{P}_{\text{FKS}}} \frac{1}{s_{ij}}, \quad (2.5)$$

such that:

$$\mathrm{d}\sigma^{(X)} = \sum_{i,j} S_{i,j}\, \mathrm{d}\sigma^{(X)}, \quad (2.6)$$

where, in this example, $\sigma^{(X)}$ represents either the Born cross-section, $\sigma^{(B)}$, the virtual correction, $\sigma^{(V)}$, or the k-factor, $\sigma^{(K)}$.

To demonstrate this partitioning effect we analyse the process $e^+e^- \to q\bar{q}g$. Here, we can isolate each of the two FKS pairs $\{qg, \bar{q}g\}$ and weight all phase-space points in the divergent regions according to the behaviour of $\mathcal{S}_{i,j}$ for each pair. The first pair corresponds to either the quark and gluon going collinear or the quark or gluon going soft. Since we cannot have soft quarks, this FKS partition only contains the singularities for the soft gluon and collinear quark and gluon. The behaviour of the FKS partition function, $S_{q,g}$ can be clearly seen in figure 2, where we observe increasingly highly weighted points as s_{qg} approaches 0.

An advantage of this method is that the interpolation between singular regions is smooth since they add together to produce the overall cross-section (see equation 2.6).[1] By weighting the matrix elements in this way, phase-space points closer to the $q||g$ singularity contribute with increasing significance to the corresponding neural network's loss during training. A similar analysis can be performed for the second FKS pair in this process.

Since the FKS pairs are ordered, the upper bound on the number of pairs for our processes is:

$$N_{\text{max}} = \frac{n_j(n_j - 1)}{2} - 1, \quad (2.7)$$

[1] An alternative implementation would be to partition the phase-space in a piecewise manner according to Heaviside step functions (as in [34]); however, this introduces an additional set of scale choices and significantly reduces the number of phase-space points left for each network to learn the complicated divergent structure. Indeed, we found that when partitioning piecewise the network performs significantly worse in comparison to this smooth implementation.

JHEP06(2020)114

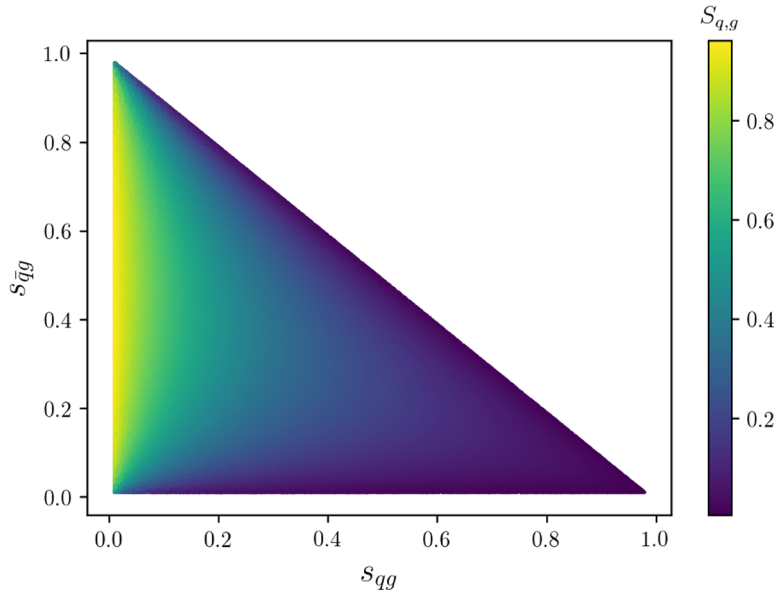

Figure 2. Behaviour of the $S_{q,g}$ FKS partition function.

where n_j is the number of jets and the -1 comes from the fact that $\{q\bar{q}\}$ is not an FKS pair by definition. It should be noted that the number of pairs can be reduced in reality due to the symmetric behaviour of all gluon-gluon, or quark-gluon pairs; however, for simplicity we partition into $N_{\max}$ regions. For example, in the example above, $N_{\max} = 2$ but $N = 1$ since the behaviours of the two pairs in this process are identical.

After using the FKS partition function to divide the region $\mathcal{R}_{\text{div}}$, we are left with $N_{\max} + 1$ regions in total across which we train the same number of networks. We find that setting the scale to $y_p = 0.01$ is generally applicable to all processes analysed.

2.2 Neural network setup

We compare the performance of two neural network setups, firstly a singular network is trained over the entire uniformly sampled phase-space, and secondly an ensemble of $N_{\max} + 1$ networks are trained over the partitioned phase-space.

2.2.1 Data

The phase-space is uniformly sampled using the RAMBO algorithm [38], with each point initially having a weighting of unity. At LO, we train the naive model on data generated from sampling over the entire phase-space uniformly, whereas we train the partitioned model on samples drawn equally from the divergent and non-divergent regions.[2] At NLO, due to the computational expense of virtual matrix element calculation, the phase-space is uniformly sampled as a whole and then divided into $\mathcal{R}_{\text{div}}$ and $\mathcal{R}_{\text{non-div}}$ regions after sampling. RAMBO was chosen for its simplicity, the ease with which it can be altered

[2]Testing was done to assess the significance of equally sampling from the divergent and non-divergent regions of phase-space when training the naive model as well, although we found little significant performance increase relative to that of using the partitioned model.

JHEP06(2020)114

to our specifications, and because it highlights interesting pitfalls and difficulties in high-dimensional functional approximations (see more on this below). In total we generate 500k phase-space points for training at LO, but only 100k at NLO due to the complexity of the problem.

The infrared poles in the matrix element result in singularities. Neural networks for classification tasks have been repeatedly shown to perform better when datasets are balanced, thus helping to avoid bias in the classification. Balancing can be done through a variety of methods such as over and under sampling, as well as loss function weightings. In regression tasks, the equivalent to class imbalances are under sampled regions that behave significantly differently to the rest of the sampled space. When doing explicit numerical calculations of the matrix element, these imbalances are not such an issue and their effect when calculating observables can be estimated by the Monte Carlo error and by phase-space resampling, yet they become significant when training a network.[3] Through balancing the training datasets in the divergent and non-divergent regions, and using the FKS partitioning method as outlined above, we hope to address the issue of underrepresented regions.

Increasingly sophisticated non-machine learning based methods for phase-space sampling using techniques such as adaptive methods, integrand factorisation and recursive stratified sampling [39–45] have been developed. Similarly, importance sampling methodologies specifically designed for QCD antenna generation exist to better capture these divergent regions given the physical knowledge of the pole structure [46, 47]. RAMBO, however, is indifferent to these variational differences in phase-space, giving a more naive sampling, yet the ability to construct an interpolation function from a uniformly sampled phase-space means we save computational time during the sampling stage. Although performance of our approximation may be increased using these more sophisticated methods, demonstrating sufficiently good results while requiring only the use of simple sampling techniques like RAMBO further shows the power of our method and the additional time savings it can offer.

Once the phase-space points are generated, we use NJET [7] to calculate the corresponding squared matrix elements at LO, and the virtual correction terms at NLO, for $e^+e^- \to Z^*/\gamma \to q\bar{q} + n_g$. We calculate all quantities in the four-dimensional helicity (FDH) scheme, assuming all external legs to be massless, with the number of light quark flavours set to $n_f = 5$, and use the same renormalisation scale as in [48] (see section 2.3).

When training the network, the dataset is split in an 80:20 ratio for training and validation. Furthermore, independently generated, unseen datasets are used for testing the performance of our models. Model testing consists of inferring on these unseen test points to create cross-section and differential plots as shown in section 3. Through generating many more points for testing than training we demonstrate the performance of our methodology as an interpolation function by further extrapolating into the divergent region.

To avoid the problem of vanishing/exploding gradients, we standardise our data to zero mean and unit variance at each input node and across the targets.

[3] We define Monte Carlo error at the per-bin level to be $\text{MC} = \sqrt{\frac{\langle f^2\rangle - \langle f\rangle^2}{N}}$, where angled brackets denote the average over the values of the function contained inside at the bin-level, and f are the matrix elements. Note that since we are using RAMBO the phase-space volume is unity.

JHEP06(2020)114

2.2.2 Architecture

Choosing an optimal network architecture is non-trivial due to the large number of parameters that can be tuned to an array of criteria. It is common to approach a singular problem using a neural network and thus optimise the architecture for that process; however, while we want to demonstrate the ability of networks to become sophisticated multi-parameter interpolation functions, we require these models to generalise to a variety of processes.

For better generalisability we do not fine-tune a network to any particular process, but rather attempt to employ the same architecture for each process. The neural networks are parameterised using Keras [49] with a Tensorflow [50] backend and comprise of fully-connected layers with an input layer of $(n_j - 1) \times 4$ nodes and output of 1, with three hidden layers made up of 20-40-20 nodes. The hidden layers all use hyperbolic tangent activation functions and the output node has a linear activation function.

The loss function is taken to be the mean squared error,

$$L = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2, \tag{2.8}$$

where n is the number of training points, $f : \mathbb{R}^d \to \mathbb{R}$ is the function describing the neural network, x_i is the ith d-dimensional input data, and y_i the corresponding target variable. The network is optimised using Adam optimisation [51], while the number of training epochs is determined through Early Stopping (see section 8.1.2 [52]), tracking the validation loss with no minimum change requirements. We recognise that by using a validation set containing only 20% of the original training set, we may be severely limiting the number of points in the increasingly divergent regions, thus skewing our Early Stopping criteria to the less divergent regions. In an attempt to mitigate this, we train with a patience of 100 epochs to measure effects in the loss function significantly later in the training regime; however, at NLO we found that this makes minimal difference to the total loss and so can be reduced to speed up network training.

The inputs to the network are the 4-momenta of $n_j - 1$ jets. Since we fix the centre-of-mass energy for training, we sought to reduce the number of input nodes for more efficient learning. We note that further reductions in the number of input parameters could be made, yet in testing this had no significant effect on performance.

2.3 Uncertainty analysis

The subject of error and uncertainty analysis in machine learning processes is receiving increasing attention (see [53, 54] and the references therein), especially in the particle physics community [55–58], yet too frequently a demonstration of rigorous error analysis in machine learning regression processes is lacking.

As stated in [53], the main sources of error arise from approximation, aleatoric and epistemic uncertainties.[4] Since we are using deep neural networks, and have tested both deeper

[4]Approximation uncertainty arises due to the model being too simplistic to allow for complex functional fitting, e.g. too few nodes or hidden layers in a neural network meaning the model isn't able to fit sufficiently non-linear functions. Aleatoric uncertainty accounts for fluctuations in the data distribution e.g. from

JHEP06(2020)114

and shallower architectural designs, we assume that errors associated with approximation uncertainties are negligible. Additionally, we do not consider aleatoric uncertainties here since our data has been generated through high-precision numerical methods. Moreover, NJET accuracy tests have been performed to measure the stochasticity in matrix element generation and found this fluctuation to be negligible.[5] Following [55] we apply similar methods highlighted for use in classification networks, to this regression task. Specifically, we focus on the measurement of precision/optimality errors which include those arising due to epistemic uncertainties.

We measure model parameter initialisation dependence by training an ensemble of models on fixed training datasets while randomly reinitialising the weights of each model. Depending on the observable, the standard deviation in the bins can be measured. Additionally, when sampling the phase-space, the Monte Carlo error is calculated; however, this does not fully account for the uncertainty in phase-space completeness. For this we bootstrap the training data thereby resampling the phase-space multiple times and training an ensemble of models with each model trained on a different dataset, while keeping the weight initialisations fixed. Since in this paper we are comparing neural network output against NJET results, to avoid the double counting of errors we only include Monte Carlo errors on the NJET results. When using models 'in production', Monte Carlo error can be added to the model uncertainty, as specified above, for a full uncertainty estimate. We note that the best possible achievable accuracy would correspond to the Monte Carlo error on the NJET result.

The performance of our methodologies are also dependent on the test set chosen. For this we quote the Monte Carlo error, although it should be noted that the same issue with determining sampling completeness occurs here. Due to the computational expense of repeated generation of test sets we do not perform this, although the uncertainty bands on the neural network approximations should be sufficient to provide evidence of our methodology since these additional dataset dependancies are negligible given the large number of test points used and the relative size of the computed Monte Carlo error compared to the model uncertainties (see section 3).

The errors on the models that we calculate are therefore the error due to model initialisation dependence and error due to the size of the training dataset, which are added in quadrature. As noted by Nachman [55], additional sources of uncertainty are inherent in the network approximation that are hard to calculate explicitly, such as dependence on the model architecture (e.g. the number of hidden layers, nodes in each layer and the types activation functions used). Due to the size of the other errors mentioned, and the lack of currently available tools for their calculation, we do not attempt to incorporate errors arising from these uncertainties into our analysis. We quote Monte Carlo error only for

measurement errors, and cannot be decreased by collecting more data from the same experimental setup. Epistemic uncertainties, on the other hand, account for uncertainties in the model, including lack of sufficient coverage of the data.

[5] NJET accuracy tests are performed by inferring on each phase-space point twice and checking the difference in the results. The threshold is set to the default value of 10^{-5} and errors arise due to lack of floating point precision and rounding errors.

JHEP06(2020)114

the testing dataset with the exception for the NLO 5-jet case in which we quote both the Monte Carlo and model errors (see figure 9).

When presenting our results, we calculate the mean of the ensemble of models trained and quote the standard error on the mean. Throughout this paper, we choose to train 20 models for each ensemble, however, this number was chosen in a slightly *ad hoc* manner, since it gave a reasonable distribution of models, and should not be interpreted as a requirement.

Theoretical uncertainties are also prevalent in all of these calculations due to variability in setting the renormalisation scale, μ. Such uncertainties propagate though the networks since a model will learn to fit data at a certain scale. In this paper we train on data generated at a fixed scale as used in [48]. We perform the normal *ad hoc* scale variation of $\mu/2$ and 2μ purely to determine the dependence of our methodology on such a scale choice. In doing so we found that the models are able to approximate the matrix elements at each scale equally well to within Monte Carlo error, and we therefore assert that model performance is not highly dependent on the value of μ in the range we analysed. Moreover, since the goal of this work is not to calculate the cross-section or k-factors of a new process, but to provide tools for estimating such values for already known process, we do not quote these as uncertainties in our methodology.

3 Results

We test our methodology on estimating both LO cross-sections and k-factors for processes up to $e^+e^- \to$ 5-jets. In addition, various observables are plotted to demonstrate the applicability of our methodology to real calculations. In general, we see that neural network approximations demonstrate wide applicability to the cases investigated, with the FKS partitioning method giving more accurate and stable results due better approximating the infrared singularities.

It should be noted that to retain consistency between the training and testing phases, we sample both datasets in the same way. The data used for training and testing the naive models has been generated using RAMBO over the entire phase-space. In contrast, the data for training and testing the partitioned models has been generated by first splitting the phase-space into divergent and non-divergent regions and uniformly sampling equally and independently in each. More details on this can be found in section 2.1. Test data here refers to the data used to create the distributions presented in this section and does not contain any phase-space points used during the training phase. Moreover, when comparing model performance against that of NJET, both the models and NJET have been used to evaluate the same phase-space points. Throughout all tests, phase-space cuts at y_{cut} are used to regulate the infrared divergences.

3.1 Approximations at LO

Although leading order calculations are not significantly computationally expensive, they pose interesting test cases for neural network approximations of high multiplicity processes with many scales and complex infrared singularity structures. Moreover, we find that

JHEP06(2020)114

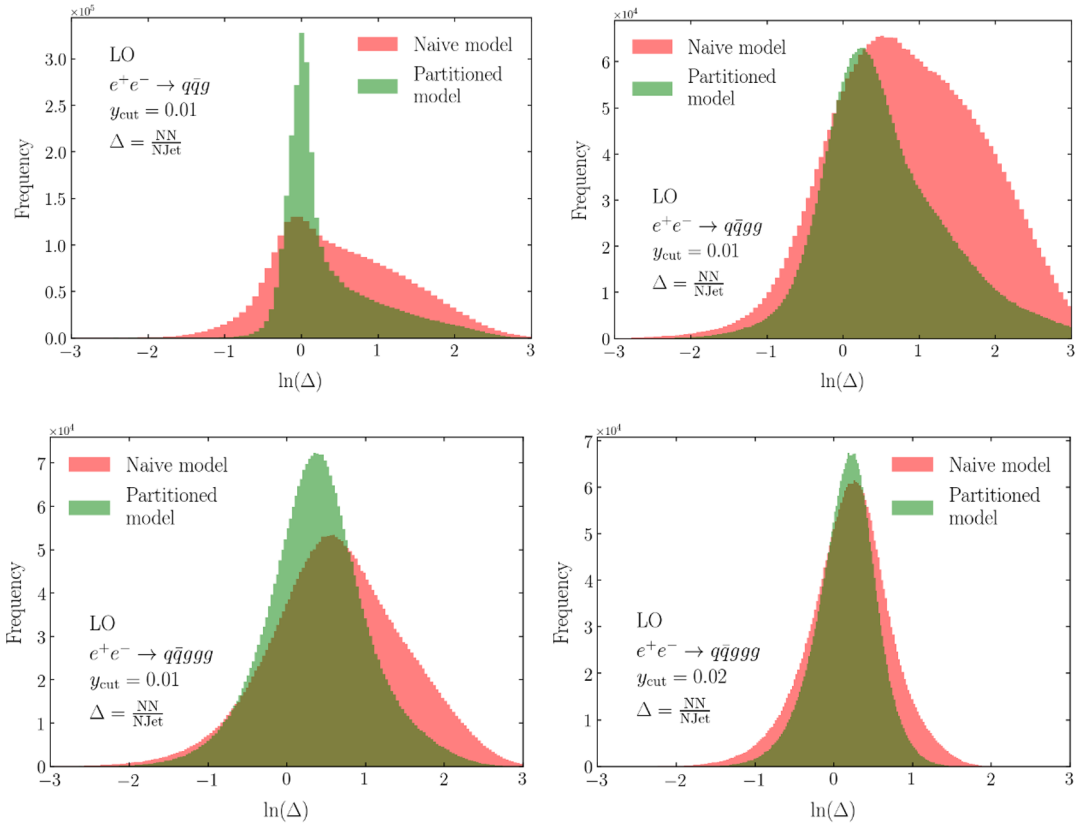

Figure 3. Born matrix element output of the naive approach (red) and partitioned approach (green) compared to the NJET calculation at different jet multiplicities and/or y_{cut} values. Outputs are taken as the average over 20 trained models.

much of what can be learnt from the performance of the models here can be applied to the NLO case.

As detailed above, we compare the naive approach where a single network is trained over the entire phase-space with the partitioned approach where an ensemble of networks trained on $N_{\text{max}} + 1$ partitions of phase-space. In determining the appropriate value of the global phase-space cut parameters, y_{cut}, we evaluate the performance of our models by calculating the ratio of the output to the NJET calculation as well as the model's ability to approximate the cross-section and different distributions.

Figure 3 shows the distribution of the neural network errors by calculating the ratio of the model output and the NJET result at each phase-space point in the test set. Since the partitioned approach gives much narrower and more Gaussian shaped distributions than the naive approach, we can clearly see that this method is preferable at the level of per-point accuracy. Additionally, the error distributions of the partitioned approach are also more closely centred on zero, in comparison to the naive approximation, thus suggesting that the partitioned model will also produce a better overall average performance as well. Note that these plots to not contain any information about the relative uncertainties attached to these model outputs, which we will discuss below.

JHEP06(2020)114

While the error plots demonstrate the per-point performance of the models, we also wish to compare their performance in calculating physics observables while also taking into account uncertainty in the data and the model setup. Figure 4 shows the approximated cross-sections of the naive and partitioned approaches as compared to those computed from the NJET matrix elements. As expected, we see a harsher y_{cut} value at 5-jets better regulates the divergent regions, thus improving both the naive and partitioned approaches; however, this harsher cut is not fully necessary in the partitioned case as the NJET result sits on the edge of the neural network uncertainty bands.

When approximating the cross-section, we find the uncertainty bands have very little noise and follow the shape of the average result closely. Since each trained network will aim to minimise the value of the loss function, and no network will perfectly learn the target distribution, for each model there will be an offset between the final trained model result and the true distribution result. (When using mean squared error, the loss function will be minimised at the average of the target distribution, although this value is unlikely to ever be achieved (see appendix A).) Since the cross-section is proportional to the average over the phase-space, for any value of n, these differences will average out such that the offsets manifest themselves as a distance away from the cross-section as calculated by NJET:

$$\frac{1}{n}\sum_{i=1}^{n}(f(x_i)-y_i) = \sigma_P - \sigma_N \tag{3.1}$$

$$= \epsilon + \mathcal{O}(\theta), \tag{3.2}$$

where σ_P and σ_N are the predicted and NJET calculated cross-section values respectively, ϵ is a fixed offset from the true cross-section and θ a small noise parameter. This therefore explains the relatively fixed distance between the model uncertainty upper and lower bounds and the NJET result.

Another result of equation (3.2) is that, unlike Monte Carlo error, inferring on more test points will not reduce the model uncertainties since such a model cannot contain more information than the training dataset has provided it. These uncertainties are intrinsically tied to the training set and the model initialisation and so any efforts to reduce errors arising at test time should therefore be focussed on addressing such uncertainties. We demonstrate an example of this by developing our partitioned method rather than focussing on changes to the test dataset.

In general, the global cuts required for the partitioned approach to be within the Monte Carlo error of the true cross-section are $\sim y_{\text{cut}} = 0.01$. These cut values are reasonable for our definition of y_{ij} and are equivalent to the cuts made in [9].

After cuts have been made, we see that the partitioned approach has a significantly reduced standard error when compared with the naive approach, with a predicted mean closer to the final stable cross-section. This difference in uncertainty can be understood by comparing the relative standard deviations of the naive model's single network and the deviations in the different networks making up the partitioned model, as we shall now show.

JHEP06(2020)114

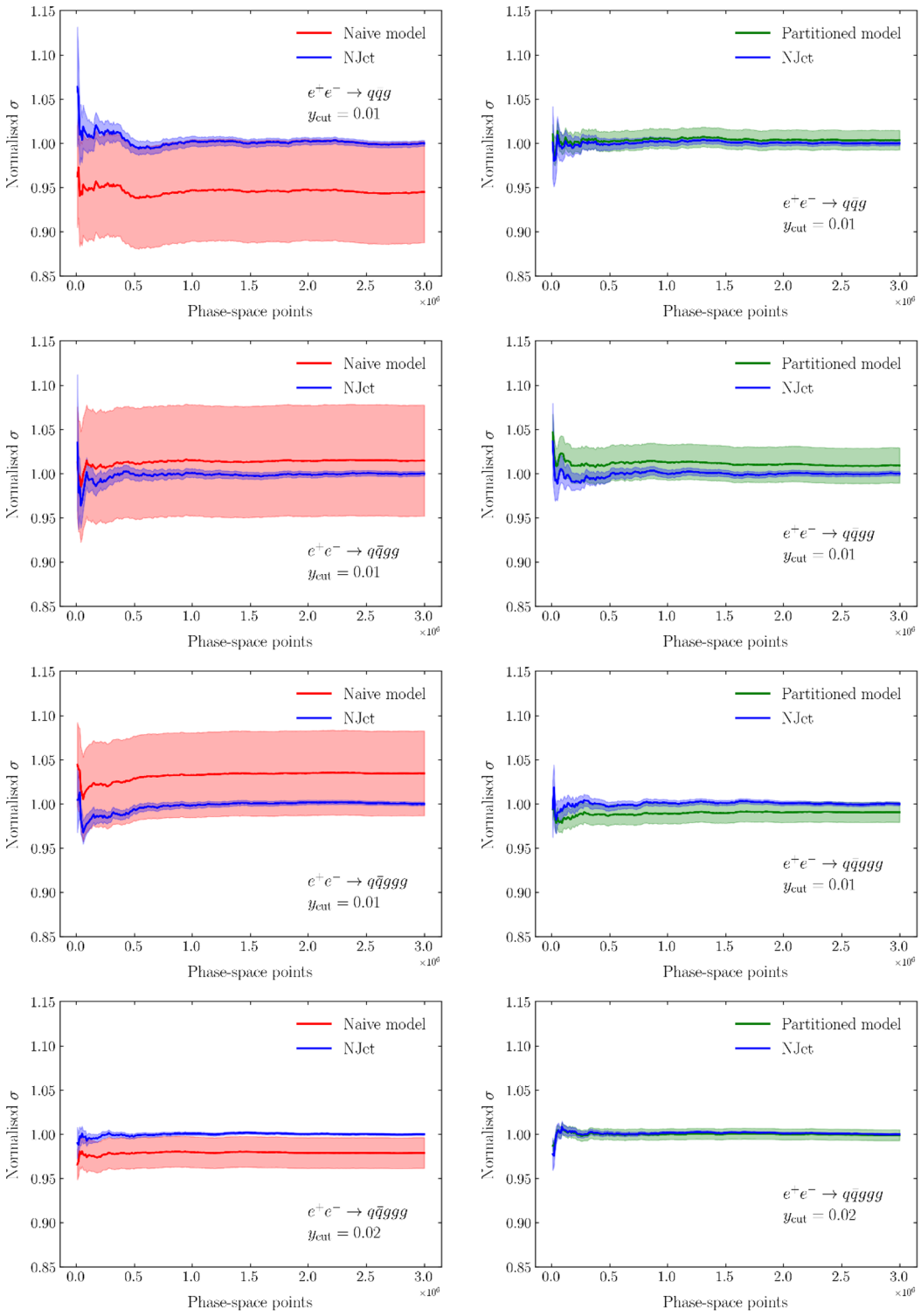

Figure 4. Comparison of the naive approach (left) vs. the partitioned approach (right) in estimating the Born normalised cross-section. Uncertainty bands denote the standard error on the mean calculated over 20 trained models (red and green) and Monte Carlo error on the NJET result (blue). We refer the reader to section 2.3 for details of the error analysis.

JHEP06(2020)114

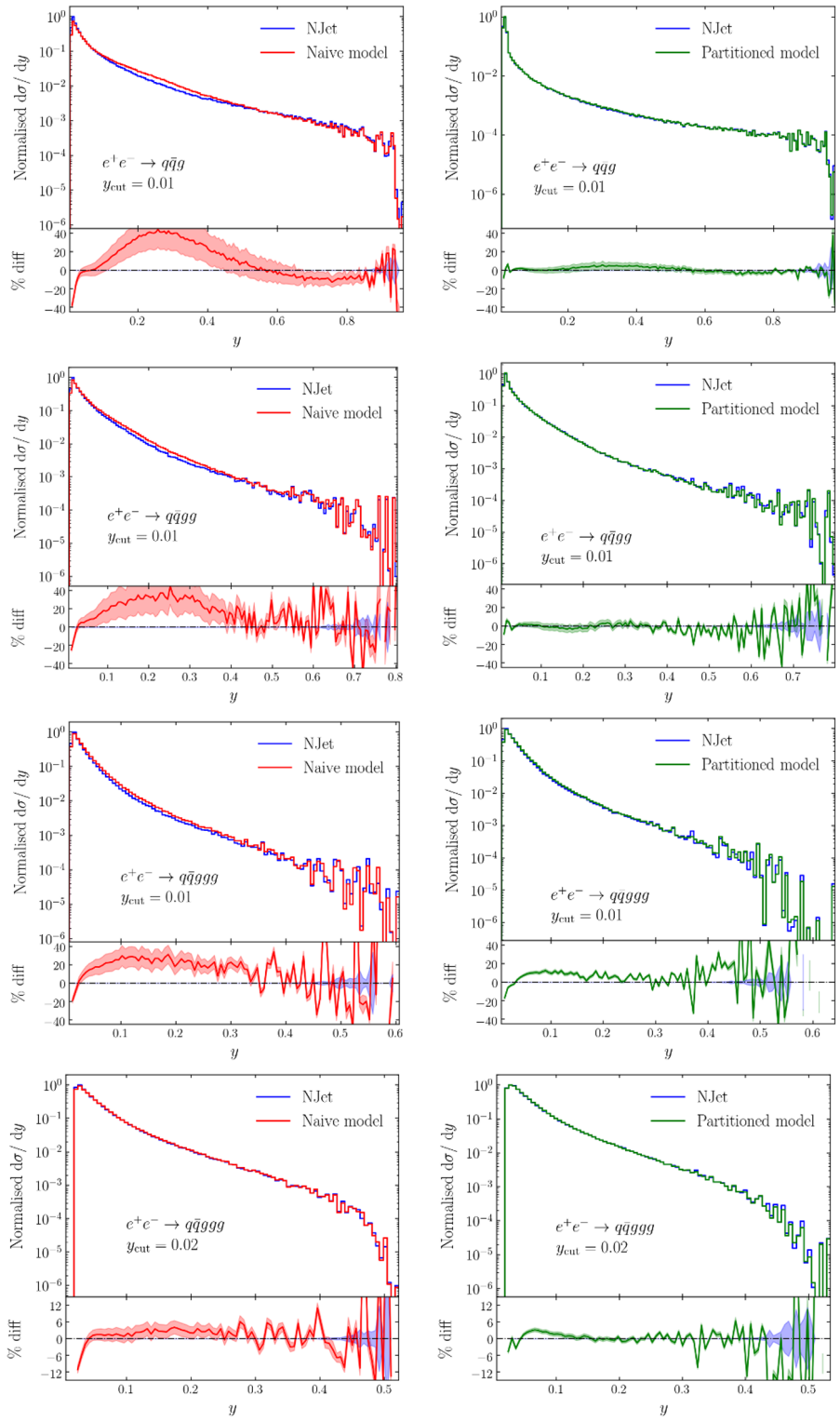

Figure 5. Comparison of the naive approach (left) vs. the partitioned approach (right) in estimating the differential cross-section against y, where y is the minimum y_{ij} as ordered by p_T. Data is normalised to the maximum NJET bin value. Uncertainty bands as described in figure 4.

JHEP06(2020)114

Let us first assume that the values of the cross-section calculated using the naive approach, σ_s, are normally distributed,[6] i.e. $\sigma_s \sim \mathcal{N}(\mu_s, \zeta_s^2)$, where μ_s is the mean of the normal distribution and ζ_s is the standard deviation. Secondly, we note that in the case of the partitioned method the outputs of the networks trained over different partitions are first summed (cf. equation (2.6)) giving:

$$\sigma_{\text{FKS}} = \sum_{p=1}^{N_{\max}} \mathrm{d}\sigma_p + \mathrm{d}\sigma_{\text{non-div}}, \tag{3.3}$$

where $N_{\max}$ is defined in equation (2.7),[7] $\mathrm{d}\sigma_p$ is the sum over all weighted matrix elements for a given FKS pair. Since we only partition the divergent region, $\mathcal{R}_{\text{div}}$, according to the FKS partition function, we add the differential cross-section over the non-divergent region, $\mathrm{d}\sigma_{\text{non-div}}$.

Given that the uncertainties in the individual networks making up the partitioned model are expected to manifest themselves in a similar way to the naive approach, we may also assume that these are drawn from a normal distribution such that:

$$\forall p \in \{1, \ldots, N_{\max}\} : \quad \mathrm{d}\sigma_p \sim \mathcal{N}(\mu_p, \zeta_p^2), \quad \mathrm{d}\sigma_{\text{non-div}} \sim \mathcal{N}(\mu_{\text{non-div}}, \zeta_{\text{non-div}}^2), \tag{3.4}$$

$$\implies \sigma_{\text{FKS}} \sim \sum_{p=1}^{N_{\max}} \mathcal{N}(\mu_p, \zeta_p^2) + \mathcal{N}(\mu_{\text{non-div}}, \zeta_{\text{non-div}}^2) \tag{3.5}$$

$$\sim \mathcal{N}\left(\sum_{p=1}^{N_{\max}} \mu_p, \sum_{p=1}^{N_{\max}} \zeta_p^2\right) + \mathcal{N}(\mu_{\text{non-div}}, \zeta_{\text{non-div}}^2) \tag{3.6}$$

$$\sim \mathcal{N}\left(\sum_{p=1}^{N_{\max}} \mu_p + \mu_{\text{non-div}}, \sum_{p=1}^{N_{\max}} \zeta_p^2 + \zeta_{\text{non-div}}^2\right) \tag{3.7}$$

$$:= \mathcal{N}(\mu_{\text{FKS}}, \zeta_{\text{FKS}}^2). \tag{3.8}$$

Since the uncertainties in the partitioned method are smaller than those found when using the naive approach:

$$\zeta_{\text{FKS}}^2 < \zeta_s^2 \tag{3.9}$$

$$\implies \zeta_p^2 < \zeta_s^2, \forall p \in \{1, \ldots, N_{\max}\} \text{ and } \zeta_{\text{non-div}}^2 < \zeta_s^2. \tag{3.10}$$

From equation (3.10) we see that not only does the partitioned method have a reduced uncertainty in comparison to the naive method, but that each individual network making

[6] This is a reasonable assumption given that we would expect the uncertainty due to initialisation and dataset size to focus around a central mean value, with greater degrees of fluctuation becoming increasingly less likely. Additionally, any difference between the mean and the NJET result would likely be systematic of the model architecture choice, sampling algorithm and other factors external to the uncertainty measured here, thus resulting in a symmetric distribution, up to an approximation.

[7] In our implementation, for future process independence and coding simplicity we actually have $N_{\max}+1$ pairs since we do not discard the $q\bar{q}$ pair. In the processes examined in this paper, this has the effect of splitting the non-divergent region into two parts although, given the ease with which the networks are able to learn this region, we do not find this causing an issue.

JHEP06(2020)114

up the partitioned model also has a reduced uncertainty, thus supporting the claim that by using the partitioned method, the networks learning the divergent structure are more certain about what they are learning and less sensitive to both model initialisation and dataset size.

The overall accuracy of the partitioned approach, combined with the implications of equation (3.10), demonstrates that we are learning the divergent structure of the amplitude sufficiently well. As discussed in section 2.3, it should be noted that figure 4 and figure 5 do not show the performance of a single model, but rather the average of 20 trained models with their equivalent standard error. Although one does not have to train this many models to still get a good approximation, in section 3.2 we will see that training additional models is computationally cheap and thus not a large hinderance.

Figure 5 shows the differential cross-section of the y_{ij} distribution of the two softest jets as ordered by p_T. Again, we plot the mean of the 20 trained models and the standard error on the mean. These differential distributions were chosen as they highlight the performance of the models in hard-to-sample regions of phase-space, in particular some of the regions we would expect the FKS partition function to assist with learning. Indeed, we see a significant improvement when using our partitioned method both, in comparison to the performance of the naive approach, in overall per-bin accuracy and stability. In addition, the partitioned method also produces narrower uncertainty bands than the naive approach, thus demonstrating its higher confidence in these regions. While this confidence is seen to be slightly misplaced in the case of the 5-jet plot at $y_{cut} = 0.01$, we see the harsher cut mostly correcting for this and producing good agreement between the NJET and partitioned results. Similar reasoning as given in equations (3.4)–(3.10) can be applied to the per-bin uncertainty differences between the naive and partitioned approaches.

Overall, the partitioned model is shown to produce more accurate and reliable results in LO approximations than the naive approach. While it can be argued that there is greater computational expense in training multiple networks, given the very low cost of network training in comparison to the data generation time this is considered to be negligible, particularly at higher orders (see section 3.2 for more details).

3.2 Virtual approximations at NLO

When approximating the k-factor, the infrared singularities present in the previous examples have been normalised. This normalisation regulates the number of large divergences in phase-space, allowing the network to focus more on learning the loop-induced divergences. Additionally, although the FKS method is especially useful for isolating soft and collinear divergences at LO, given the presence of $\log(s_{ij})$ terms in the virtual corrections we still expect to see improvements by using the partitioned method when approximating the k-factor.

As in the LO case, in figure 6 we plot the error distributions for the naive and partitioned cases by comparing the network outputs to the NJET calculations at the per-point level. In the 3 and 4-jet cases we see that both methods perform relatively similarly, with the naive approach appearing to be slightly better in the case of 4-jets. However, it should

JHEP06(2020)114

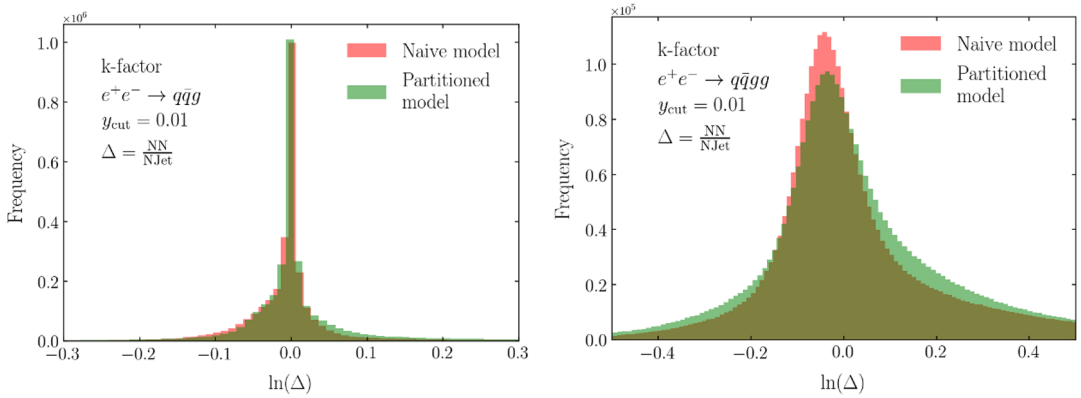

Figure 6. k-factor output of the naive approach (red) and the partitioned approach (green) compared to the NJET calculation at different multiplicities. Outputs are taken as the average over 20 trained models.

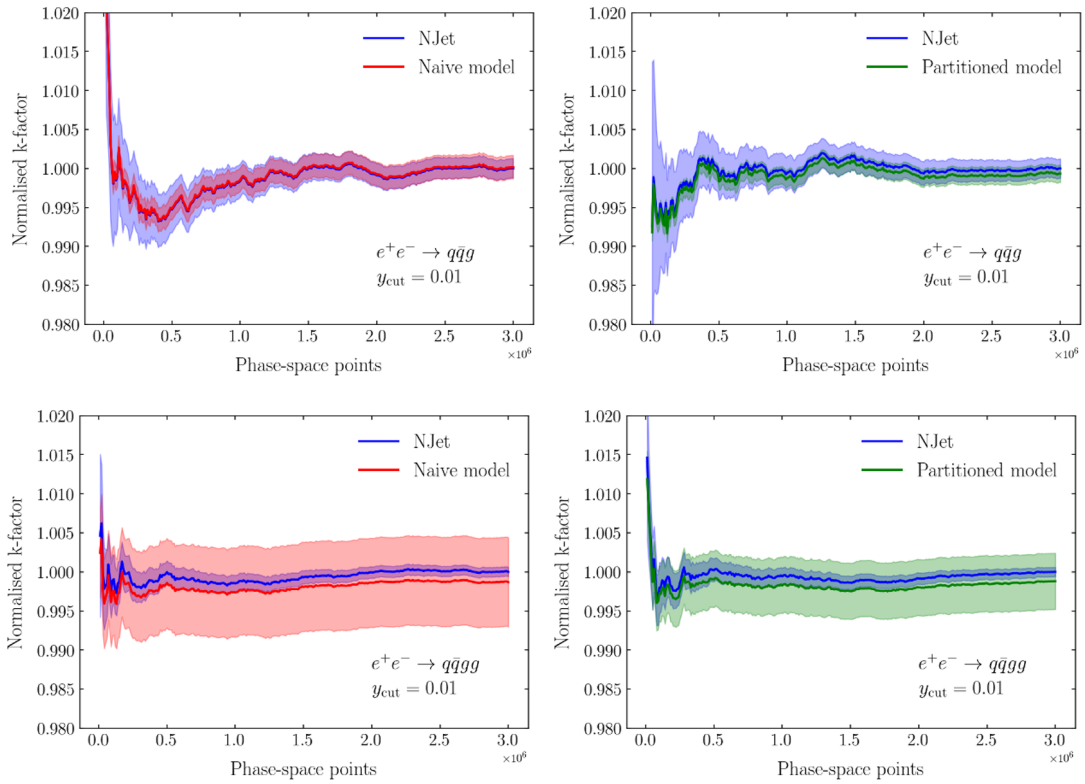

Figure 7. Comparison of the naive approach (left) vs. the partitioned approach (right) in estimating the normalised NLO/LO k-factors. Uncertainty bands as described in figure 4.

again be noted that these plots do not contain information about the network uncertainty and so should not be interpreted as the sole measure of performance.

In figure 7 we see that both the naive and the partitioned approaches approximate the k-factor to within Monte Carlo error at 3-jets, and are within the percent level at 4-jets. Although either methodology would be suitable for use, the partitioned approach requires

JHEP06(2020)114

	10k training, 1M inference			100k training, 1M inference		
	NJET	Partitioned approach		NJET	Partitioned approach	
Jets	Time (hrs)	Time (hrs)	% diff	Time (hrs)	Time (hrs)	% diff
3	13.2	0.15	-0.5 ± 0.3	13.2	1.32	0.1 ± 0.2
4	194	1.97	0.5 ± 0.5	194	19.4	0.1 ± 0.4
5	6.39×10^3	63.9	—	6.39×10^3	639	—

Table 1. Time required for k-factor calculation at different multiplicities requiring 1M points, while training on 10k and 100k points. Performance of the partitioned approach is assessed by calculating the percentage difference in the cross-section approximation normalised to the NJET result. Errors are calculated by adding the model uncertainty and Monte Carlo error from the NJET result in quadrature. These results assume all calculations take place on a single CPU core and that the training points form part of the inference set. Training on 10k points is fast but not necessarily reliable, whereas using 100k points gives more reliable results and so may be a more reasonable estimate of the speed up. Results are not given for 5-jets since we did not generate testing data at this multiplicity.

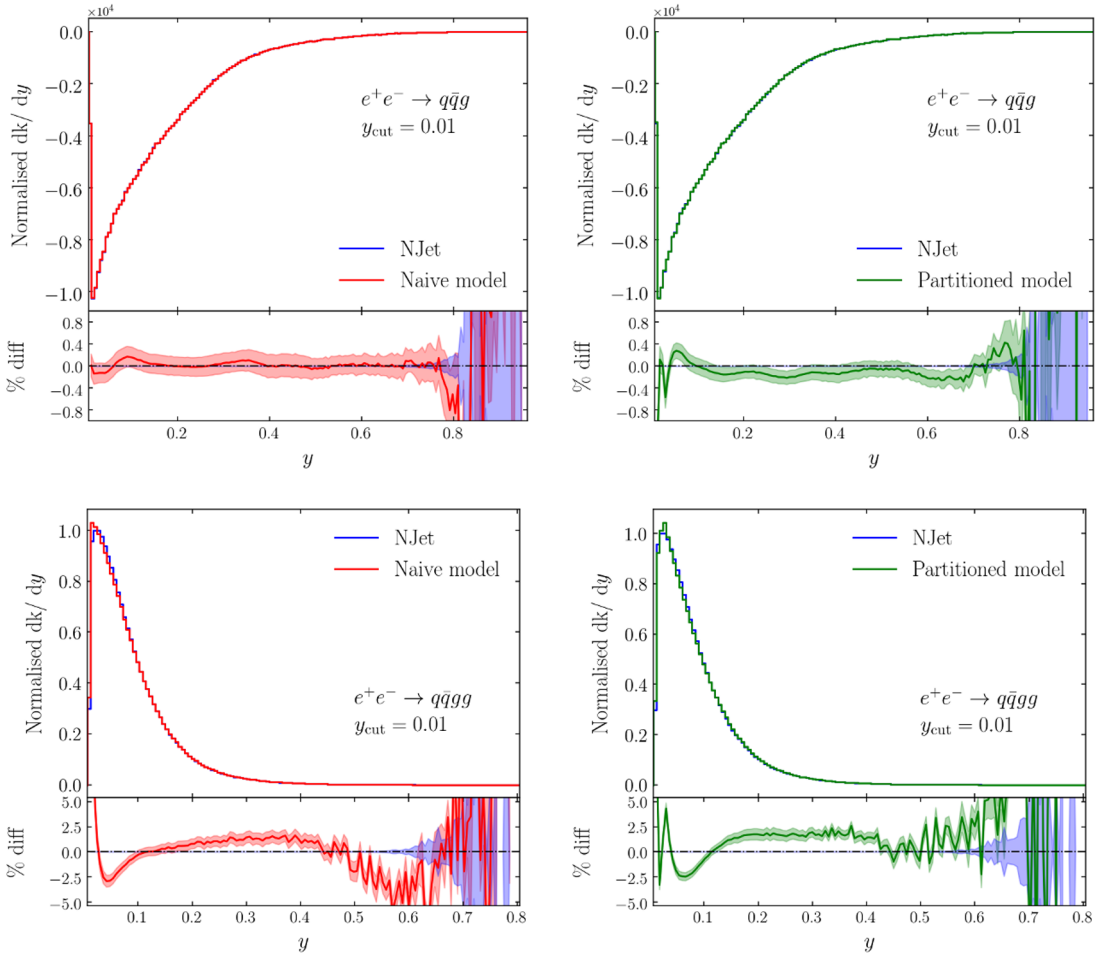

Figure 8. Comparison of the naive approach (left) vs. the partitioned approach (right) in estimating the differential NLO/LO k-factors against y, where y is the minimum y_{ij} as ordered by p_T. Data is normalised to the maximum NJET bin value. Uncertainty bands as described in figure 4.

JHEP06(2020)114

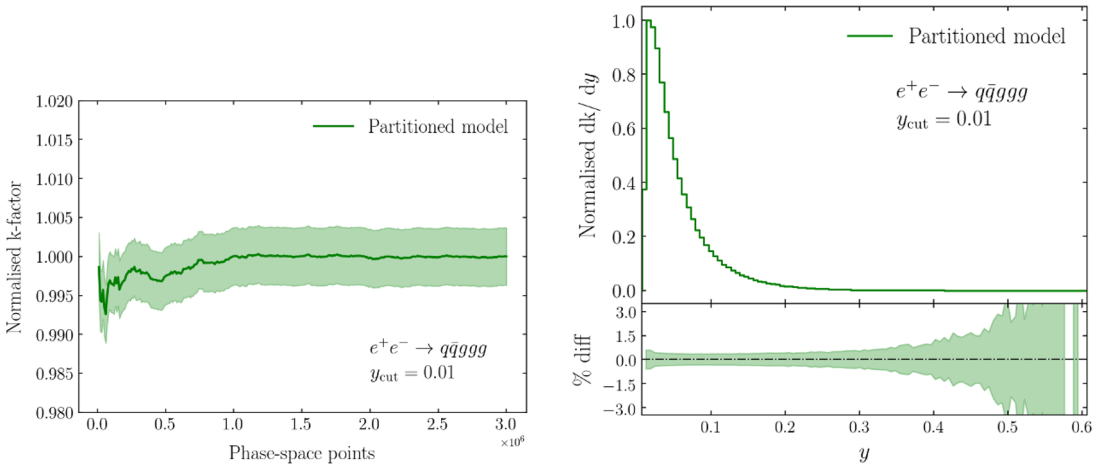

Figure 9. Normalised NLO/LO k-factor and differential k-factor against y, where y is the minimum y_{ij} as ordered by p_T, at 5 jets using just the partitioned approach. Data in the differential plot is normalised to the maximum network output value. Uncertainty bands denote the following errors added in quadrature: 1 standard error from model uncertainties and Monte Carlo error on the result itself. Uncertainty bands are given as a percentage of the mean calculated over 20 trained models.

little more computational time in comparison to the naive model, while producing narrower uncertainty bands. For robustness at higher multiplicity, the partitioned method remains the more optimal method.

A comparison between the computational speed of different methods of k-factor computation and calculation can be found in table 1. Here we see a dramatic speed up when using the network approximation as opposed to current numerical methods, with the dominant time saving coming from the reduction of the number of matrix elements having to be explicitly calculated using NJET (i.e. in the case of training on 100k points and inferring on 1M at high multiplicity the speed up is $\mathcal{O}(10)$). Moreover, the assertion that the partitioned method is not significantly more expensive than the naive approach can be verified. It should be noted that by only training on 10k points we may achieve unacceptable performance when compared to the 100k results. The results presented in the table are therefore designed to demonstrate the computational time required for network training in comparison to the NJET calculation, as opposed to providing guidelines on how many training points to use.

As in section 3.1, we plot the differential k-factors of the y distribution of the two softest jets as ordered by p_T. In figure 8 we see that both the naive, and partitioned approaches, model the data well. As before, the partitioned method provides us with slightly narrower uncertainty bands in both the 3-jet and 4-jet cases. Additionally, although neither the naive model, nor partitioned model approximate the peak in the 4-jet distribution exactly, the peak location is more accurately approximated by the partitioned approach with only a single bin at the peak being significantly ill-approximated. While we do not necessarily see much improvement in using the partitioned approach, given that the additional training time required is negligible in comparison to the data generation, as well as its performance

JHEP06(2020)114

in approximating the overall cross-section, we still see the partitioned approach as a viable and beneficial method to use for k-factor approximation. It should be noted that similar reasoning as given in equations (3.4)–(3.10) can again be applied to the k-factor and per-bin uncertainty differences between the naive and partitioned model approaches at NLO.

Finally, in the case of 5-jets we demonstrate our methodology as it may be used in practice. In figure 9 we show how one may predict on a set of points with no known NJET results for testing, while understanding the associated neural network errors. From these plots we clearly see that the partitioned method has associated errors only at the level of 0.5% in the cross-section, with larger uncertainties in the regions of the differential plot where one would expect Monte Carlo error to dominate.

As highlighted above, when you do not have a test set for comparison, it may be hard to validate the optimal number of training points required for a good approximation. While at NLO we present the results of networks trained on 100k points, and found this number to be relatively optimal with regard to accuracy, stability and training time, we do not claim that this will always be the case for other processes. Although generating more NJET matrix elements for testing is the best way to assess network accuracy, a possible substitute would be to test on the training data. While this is not generally regarded as good practice, given the problem at hand it may not be as bad as in other cases. For instance, unless there is a large degree of noise in the cross-section given the size of the training dataset, as an initial measure of model performance we can quantify the uncertainty in our training set and assess the proximity of our network uncertainties and this Monte Carlo error. Additionally, our network uncertainty calculation depends only on the network's behaviour relative to the training set and is independent of the test set. Therefore, although testing on the training set is still not ideal, given how we calculate our network uncertainties and by using our physics knowledge of the Monte Carlo error, we are able to use this as a first test of network performance without having to generate additional testing data.

4 Conclusions

In this article we have explored the possibility of optimising simulations for many scale processes needed for LHC analyses. Machine learning technology is finding an increasing number of applications in particle physics and offers the potential to dramatically reduce the CPU cost of expensive simulations.

Simulations are often made expensive by the high cost of calculating scatting amplitudes. The number of times these amplitudes are evaluated for a given process is determined by the integration method. Recent work, such as that of [10, 11], develop novel methods for the integration of scattering cross sections. These methods have the potential to be combined with new techniques for efficient matrix element computation, such as those presented in this study, to provide even greater cost savings when calculating cross-sections and differentials.

The application to scattering amplitudes is a little different to classic examples of neural networks in that the dataset is exact.[8] We can also have complete control over the range of

[8]Technically we restrict to double precision, although higher precision arithmetic could be used in principle.

JHEP06(2020)114

the dataset, although the CPU cost of obtaining the data can be very high. The challenge is to make a sufficiently good fit to the data that a reliable interpolation and extrapolation of differential cross-sections can be made. The CPU cost of the extrapolation/interpolation is negligible in this procedure so the further the network can be extrapolated the better the computational speed up.

In this study we have looked at multi-scale amplitudes which are not well suited to more traditional approximations with polynomial grids. At one-loop scattering for $2 \to 4$ or higher multiplicity becomes extremely expensive even with modern automated tools. We find that a reliable amplitude approximation can be difficult to achieve when using a naive single neural network due to the large changes in the amplitude related to its singularity structure. We compare this naive approach to a technique in which an ensemble of networks are use to approximate the amplitude by separating the singularities using an FKS partitioning.

Understanding the reliability of this approach is one of the biggest challenges. By varying the initial data and parameter initialisations used in the network, we find a way to estimate the error on the networks. For all but the highest multiplicity, $e^+e^- \to 5$ jets, we also provide comparisons to direct integration of the amplitude. At LO we observe that the FKS partitioning provides significantly more reliable and accurate estimates than the naive approach, while in the case of NLO k-factors, where the leading order singularity structure is divided out, the partitioning still helps in these regards with results accurate to within a few %. Moreover, we show in equations (3.4)–(3.10) that each network in the partitioned model has a smaller associated uncertainty than that of the naive model, thus suggesting that the partitioned model is learning the divergent structure with a higher confidence than the naive model. Indeed, this is the case at both LO and NLO. The networks not only provide good scattering amplitude approximations, but also lead to reliable predictions with at least a factor of 10 improvement to the complete simulation.

In this initial study we have made a number of simplifications whose effect could be important when using the technique for a realistic analysis. Firstly, we employed a simple flat phase-space generation using the RAMBO algorithm. This makes it hard to compare with the more efficient generators used in state-of-the-art Monte-Carlo simulations. The JADE jet algorithm may exacerbate the soft singularities and so the effect of alternative jet algorithms, as well as the effect of introducing initial state singularities in pp collisions, should be studied in the future. We also see in the higher multiplicity cases that the error from the neural network approximation does start to increase. It may be in these cases that the NLO FKS separation requires modification. In this study we used a simple version of the partition function based only on the kinematic invariants. In general we can alter the scaling power of the invariants in the various limits which will affect the behaviour of the FKS regions away from the singularities. We may also find that effects of higher order, double unresolved singularities begin to play a role. Since NNLO sector decomposition strategies are available it would be interesting to explore this direction in the future. Another important step would be to apply the technique to integration of infrared subtracted, real radiation. This case is currently the most CPU expensive part of producing precise differential distributions for comparison with the experiments.

JHEP06(2020)114

Acknowledgments

We are very grateful to Frank Krauss, Michael Spannowsky and Daniel Mâitre for useful discussions. JB is supported by the U.K. Science and Technology Facilities Council (STFC) grant number ST/P006744/1. SB is supported by an STFC Rutherford Fellowship ST/L004925/1. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 772099.

A Average tendencies of the mean squared error

This section is heavily borrowed from the appendices of [55, 59] but is repeated here given the different applications of our specific problem.

For d-dimensional input data $X \in \mathbb{R}^d$ and targets $Y \in \mathbb{R}$, we train a network that acts as a function $f : \mathbb{R}^d \to \mathbb{R}$ by minimising a loss function, L, averaged over all points. In this domain, the distributions X and Y are clearly not independent and form a joint probability density (X, Y). The output of the neural network, given some specific input variables, which minimises the loss function averaged over the entire training dataset is given by:

$$l(x) = \text{argmin}_f(\mathbb{E}[L(f(x), Y)|X = x]), \tag{A.1}$$

where $\mathbb{E}$ is the expectation value and $\text{argmin}_g(h(g(x)))$ denotes the values of a function g that minimise h.

For the case of the mean squared error, $L(f(X), Y) = (f(X) - Y)^2$, equation (A.1) becomes:

$$\begin{aligned} l(x) &= \text{argmin}_f(\mathbb{E}[(f(x) - Y)^2|X = x]) && \text{(A.2)} \\ &= \text{argmin}_f(\mathbb{E}[f(x)^2 - 2f(x)Y + Y^2|X = x]) && \text{(A.3)} \\ &= \text{argmin}_f(f(x)^2 - 2f(x)\,\mathbb{E}[Y|X = x]). && \text{(A.4)} \end{aligned}$$

Minimising $l(x)$ now gives: $l(x) = \mathbb{E}[Y|X = x]$, thus demonstrating that the mean squared error approaches the average value of the target distribution.

B FKS pairs and partition functions

The FKS subtraction formalism was designed to provide a framework by which the divergent structure arising from the real radiation corrections at NLO can be constructed and subtracted in $(n + 1)$ phase-space, where n is the number of jets at the Born level, and added back in and solved analytically via dimensional regularisation [34]:

$$\sigma_{\text{NLO}} = \int_n \mathrm{d}\sigma^{(B)} + \int_n \left[\mathrm{d}\sigma^{(V)} + \int_1 \mathrm{d}\sigma^{(S)}\right] + \int_{n+1} [\mathrm{d}\sigma^{(R)} - \mathrm{d}\sigma^{(S)}], \tag{B.1}$$

where $\sigma^{(B)}$ is the Born cross-section, $\sigma^{(R)}$ and $\sigma^{(V)}$ are the real and virtual corrections at NLO and $\sigma^{(S)}$ is the real singular structure. By performing subtraction we are able to

JHEP06(2020)114

ensure that the singular structures of the virtual and real corrections cancel, thus leaving us with a non-divergent NLO cross-section.

For the processes considered here, the most general way of defining FKS pairs is given by:

$$\mathcal{P}_{\text{FKS}} = \{(i,j)\,|\,1 \leq\ i \leq n_g+2,\ 3 \leq\ j \leq n_g+2, i \neq j,$$
$$\mathcal{M}^{(n+1,0)} \to \infty \text{ if } p_i^0 \to 0 \text{ or } p_j^0 \to 0 \text{ or } \vec{p}_i||\vec{p}_j\}, \quad \text{(B.2)}$$

which is the equivalent definition as that used in equation (2.4), but where for our purposes we have used the pairs defined by the Born and virtual correction divergent structures since we do not calculate real corrections and we are not trying to perform subtraction.

Given that FKS pairs are ordered, there is redundancy in equation (B.2) since we will double count the soft singularities. An alternative definition is just to drop the $p_j^0 \to 0$ criteria to get:

$$\mathcal{P}_{\text{FKS}} = \{(i,j)\,|\,1 \leq\ i \leq n_g+2,\ 3 \leq\ j \leq n_g+2, i \neq j,$$
$$\mathcal{M}^{(n+1,0)} \to \infty \text{ if } p_i^0 \to 0 \text{ or } \vec{p}_i||\vec{p}_j\}, \quad \text{(B.3)}$$

as shown in [35]. By using the definition given in equation (B.3), we end up with the general FKS criteria that *each FKS partition contain at most one collinear and one soft singularity*. Formalising this mathematically allows us to require the following criteria be met by any such FKS partition function, $\mathcal{S}_{i,j}$ (adapted from [35]):

$$\sum_{(i,j)\in\mathcal{P}_{\text{FKS}}} \mathcal{S}_{i,j} = 1, \quad \text{(B.4)}$$

$$\lim_{\vec{p}_k||\vec{p}_l} \mathcal{S}_{i,j} = 0,\ \forall(k,l) \in \mathcal{P}_{\text{FKS}} \text{ with } (k,l) \neq (i,j), \quad \text{(B.5)}$$

$$\lim_{p_k^0\to 0} \mathcal{S}_{i,j} = 0,\ \forall(k,l) \in \mathcal{P}_{\text{FKS}} \text{ with } k \neq i. \quad \text{(B.6)}$$

Examples of partition functions satisfying these conditions are given in [35] in terms of energies and angles and in [60] in terms of s_{ij} variables among others.

While defining a function in terms of energies and angles can be beneficial when doing full FKS subtraction, for ease of computation we use the Lorentz invariant s_{ij} variables defined in equation (2.5). However, we note that our definition of the FKS partition function does not satisfy equation (B.6) and therefore some of our partitions will contain multiple soft singularities and thus result in redundancies.

References

[1] M. Czakon, *Tops from light quarks: full mass dependence at two-loops in QCD*, *Phys. Lett.* **B 664** (2008) 307 [arXiv:0803.1400] [INSPIRE].

JHEP06(2020)114

[2] S. Borowka et al., *Higgs boson pair production in gluon fusion at next-to-leading order with full top-quark mass dependence*, *Phys. Rev. Lett.* **117** (2016) 012001 [*Erratum ibid.* **117** (2016) 079901] [arXiv:1604.06447] [INSPIRE].

[3] G. Heinrich et al., *NLO predictions for Higgs boson pair production with full top quark mass dependence matched to parton showers*, *JHEP* **08** (2017) 088 [arXiv:1703.09252] [INSPIRE].

[4] S.P. Jones, M. Kerner and G. Luisoni, *Next-to-leading-order QCD corrections to Higgs boson plus jet production with full top-quark mass dependence*, *Phys. Rev. Lett.* **120** (2018) 162001 [arXiv:1802.00349] [INSPIRE].

[5] G. Heinrich et al., *Probing the trilinear Higgs boson coupling in di-Higgs production at NLO QCD including parton shower effects*, *JHEP* **06** (2019) 066 [arXiv:1903.08137] [INSPIRE].

[6] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, *Math. Control. Sign. Syst.* **2** (1989) 303.

[7] S. Badger, B. Biedermann, P. Uwer and V. Yundin, *Numerical evaluation of virtual corrections to multi-jet production in massless QCD*, *Comput. Phys. Commun.* **184** (2013) 1981 [arXiv:1209.0100] [INSPIRE].

[8] J. Bendavid, *Efficient Monte Carlo integration using boosted decision trees and generative deep neural networks*, arXiv:1707.00028 [INSPIRE].

[9] M.D. Klimek and M. Perelstein, *Neural network-based approach to phase space integration*, arXiv:1810.11509 [INSPIRE].

[10] E. Bothmann et al., *Exploring phase space with neural importance sampling*, *SciPost Phys.* **8** (2020) 069 [arXiv:2001.05478] [INSPIRE].

[11] C. Gao et al., *Event generation with normalizing flows*, *Phys. Rev.* **D 101** (2020) 076002 [arXiv:2001.10028] [INSPIRE].

[12] L. Dinh, D. Krueger and Y. Bengio, *NICE: Non-linear independent components estimation*, in the proceedings of 3rd *International Conference on Learning Representations (ICLR 2015)*, May 7–9, San Diega, U.S.A. (2015).

[13] S. Otten et al., *DeepXS: Fast approximation of MSSM electroweak cross sections at NLO*, *Eur. Phys. J.* **C 80** (2020) 12 [arXiv:1810.08312] [INSPIRE].

[14] I. Goodfellow et al., *Generative adversarial nets*, in *Advances in neural information processing systems 27*, Z. Ghahramani et al. eds., Curran Associates Inc., U.S.A. (2014),

[15] S. Otten et al., *Event generation and statistical sampling for physics with deep generative models and a density information buffer*, arXiv:1901.00875 [INSPIRE].

[16] B. Hashemi et al., *LHC analysis-specific datasets with generative adversarial networks*, arXiv:1901.05282 [INSPIRE].

[17] R. Di Sipio, M. Faucci Giannelli, S. Ketabchi Haghighat and S. Palazzo, *DijetGAN: a generative-adversarial network approach for the simulation of QCD dijet events at the LHC*, *JHEP* **08** (2020) 110 [arXiv:1903.02433] [INSPIRE].

[18] A. Butter, T. Plehn and R. Winterhalder, *How to GAN LHC events*, *SciPost Phys.* **7** (2019) 075 [arXiv:1907.03764] [INSPIRE].

[19] A. Butter, T. Plehn and R. Winterhalder, *How to GAN event subtraction*, arXiv:1912.08824 [INSPIRE].

[20] S. Carrazza and F.A. Dreyer, *Lund jet images from generative and cycle-consistent adversarial networks*, *Eur. Phys. J.* **C 79** (2019) 979 [arXiv:1909.01359] [INSPIRE].

JHEP06(2020)114

[21] SHiP collaboration, *Fast simulation of muons produced at the SHiP experiment using generative adversarial networks*, 2019 *JINST* **14** P11028 [arXiv:1909.04451] [INSPIRE].

[22] F. Bishara and M. Montull, *(Machine) learning amplitudes for faster event generation*, arXiv:1912.11055 [INSPIRE].

[23] J. Bullock, *n3jet*, https://github.com/JosephPB/n3jet, (2020).

[24] G. Ossola, C.G. Papadopoulos and R. Pittau, *Reducing full one-loop amplitudes to scalar integrals at the integrand level*, *Nucl. Phys.* **B 763** (2007) 147 [hep-ph/0609007] [INSPIRE].

[25] Z. Bern, L.J. Dixon, D.C. Dunbar and D.A. Kosower, *Fusing gauge theory tree amplitudes into loop amplitudes*, *Nucl. Phys.* **B 435** (1995) 59 [hep-ph/9409265] [INSPIRE].

[26] R. Britto, F. Cachazo and B. Feng, *Generalized unitarity and one-loop amplitudes in $N = 4$ super-Yang-Mills*, *Nucl. Phys.* **B 725** (2005) 275 [hep-th/0412103] [INSPIRE].

[27] R.K. Ellis, W.T. Giele and Z. Kunszt, *A numerical unitarity formalism for evaluating one-loop amplitudes*, *JHEP* **03** (2008) 003 [arXiv:0708.2398] [INSPIRE].

[28] W.T. Giele, Z. Kunszt and K. Melnikov, *Full one-loop amplitudes from tree amplitudes*, *JHEP* **04** (2008) 049 [arXiv:0801.2237] [INSPIRE].

[29] D. Forde, *Direct extraction of one-loop integral coefficients*, *Phys. Rev.* **D 75** (2007) 125019 [arXiv:0704.1835] [INSPIRE].

[30] C.F. Berger et al., *An automated implementation of on-shell methods for one-loop amplitudes*, *Phys. Rev.* **D 78** (2008) 036003 [arXiv:0803.4180] [INSPIRE].

[31] S.D. Badger, *Direct extraction of one loop rational terms*, *JHEP* **01** (2009) 049 [arXiv:0806.4600] [INSPIRE].

[32] F.A. Berends and W.T. Giele, *Recursive calculations for processes with n gluons*, *Nucl. Phys.* **B 306** (1988) 759 [INSPIRE].

[33] T. Binoth et al., *A proposal for a standard interface between Monte Carlo tools and one-loop programs*, *Comput. Phys. Commun.* **181** (2010) 1612 [arXiv:1001.1307] [INSPIRE].

[34] S. Frixione, Z. Kunszt and A. Signer, *Three jet cross-sections to next-to-leading order*, *Nucl. Phys.* **B 467** (1996) 399 [hep-ph/9512328] [INSPIRE].

[35] R. Frederix, S. Frixione, F. Maltoni and T. Stelzer, *Automation of next-to-leading order computations in QCD: the FKS subtraction*, *JHEP* **10** (2009) 003 [arXiv:0908.4272] [INSPIRE].

[36] M. Czakon and D. Heymes, *Four-dimensional formulation of the sector-improved residue subtraction scheme*, *Nucl. Phys.* **B 890** (2014) 152 [arXiv:1408.2500] [INSPIRE].

[37] JADE collaboration, *Experimental studies on multi-jet production in e^+e^- annihilation at PETRA energies*, *Z. Phys.* **C 33** (1986) 23 [INSPIRE].

[38] R. Kleiss, W.J. Stirling and S.D. Ellis, *A new Monte Carlo treatment of multiparticle phase space at high-energies*, *Comput. Phys. Commun.* **40** (1986) 359 [INSPIRE].

[39] J.H. Friedman, M.H. Wright, *An adaptive importance sampling procedure*, Stanford University, U.S.A. (1981).

[40] G.P. Lepage, *A new algorithm for adaptive multidimensional integration*, *J. Comput. Phys.* **27** (1978) 192 [INSPIRE].

[41] G.P. Lepage, *VEGAS: an adaptive multidimensional integration program*, CLNS-80/447 (1980).

JHEP06(2020)114

[42] W.H. Press and G.R. Farrar, *Recursive stratified sampling for multidimensional Monte Carlo integration*, *Comp. Phys.* **190** (1990) 4.

[43] T. Ohl, *Vegas revisited: adaptive Monte Carlo integration beyond factorization*, *Comput. Phys. Commun.* **120** (1999) 13 [hep-ph/9806432] [INSPIRE].

[44] S. Jadach, *Foam: a general purpose cellular Monte Carlo event generator*, *Comput. Phys. Commun.* **152** (2003) 55 [physics/0203033] [INSPIRE].

[45] K. Kroeninger, S. Schumann and B. Willenberg, $(MC)^3$ – *a Multi-Channel Markov Chain Monte Carlo algorithm for phase-space sampling*, *Comput. Phys. Commun.* **186** (2015) 1 [arXiv:1404.4328] [INSPIRE].

[46] P.D. Draggiotis, A. van Hameren and R. Kleiss, *SARGE: an algorithm for generating QCD antennas*, *Phys. Lett.* **B 483** (2000) 124 [hep-ph/0004047] [INSPIRE].

[47] A. van Hameren and C.G. Papadopoulos, *A hierarchical phase space generator for QCD antenna structures*, *Eur. Phys. J.* **C 25** (2002) 563 [hep-ph/0204055] [INSPIRE].

[48] R. Frederix, S. Frixione, K. Melnikov and G. Zanderighi, *NLO QCD corrections to five-jet production at LEP and the extraction of* $\alpha_s(M_Z)$, *JHEP* **11** (2010) 050 [arXiv:1008.5313] [INSPIRE].

[49] F. Chollet et al., *Keras*, https://github.com/fchollet/keras (2015).

[50] M. Abadi et al., *TensorFlow: large-scale machine learning on heterogeneous systems*, https://www.tensorflow.org/ (2015).

[51] D.P. Kingma and J. Ba, *Adam: a method for stochastic optimization*, arXiv:1412.6980 [INSPIRE].

[52] I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*. MIT Press, U.S.A. (2016).

[53] N. Tagasovska and D. Lopez-Paz, *Single-model uncertainties for deep learning*, *NeurIPS* (2019) [arXiv:1811.00908].

[54] Y. Gal, *Uncertainty in deep learning*, Ph.D. thesis, University of Cambridge, Cambridge U.K. (2016).

[55] B. Nachman, *A guide for deploying Deep Learning in LHC searches: how to achieve optimality and account for uncertainty*, arXiv:1909.03081 [INSPIRE].

[56] B. Nachman and C. Shimmin, *AI safety for high energy physics*, arXiv:1910.08606 [INSPIRE].

[57] S. Bollweg et al., *Deep-learning jets with uncertainties and more*, *SciPost Phys.* **8** (2020) 006 [arXiv:1904.10004] [INSPIRE].

[58] C. Englert, P. Galler, P. Harris and M. Spannowsky, *Machine learning uncertainties with adversarial neural networks*, *Eur. Phys. J.* **C 79** (2019) 4 [arXiv:1807.08763] [INSPIRE].

[59] K. Cranmer, J. Pavez and G. Louppe, *Approximating likelihood ratios with calibrated discriminative classifiers*, arXiv:1506.02169 [INSPIRE].

[60] S. Frixione, E. Laenen, P. Motylinski and B.R. Webber, *Single-top production in MC@NLO*, *JHEP* **03** (2006) 092 [hep-ph/0512250] [INSPIRE].

JHEP06(2020)114